UCRL-JC-125363
PREPRINT

Parallel Domain Decomposition Methods in Fluid Models with Monte Carlo Transport

H. J. Alme
G. H. Rodrigue
G. B. Zimmerman

This paper was prepared for submittal to the
8th SIAM Conference on Parallel
Processing Scientific Computing
Minneapolis, MN
March 14-17, 1997

December 13, 1996

Parallel Domain Decomposition Methods in Fluid Models with Monte Carlo Transport[*]

Henry J. Alme[†] Garry H. Rodrigue [†] George B. Zimmerman [‡]

Abstract

To examine the domain decomposition of a coupled Monte Carlo - finite element calculation, it is important to use a domain decomposition that is suitable for the individual models. We have developed a code that simulates a Monte Carlo calculation running on a massively parallel processor. This code is used to examine the load balancing behavior of three domain decomposition strategies for a Monte Carlo calculation. The results are presented.

1 Introduction

This paper reports on results obtained for different domain decomposition strategies for parallel processing of numerical simulations of coupled Monte Carlo - finite element models. Specifically, we are examining the parallel processing of coupled fluid-transport problems in dynamic optics. One application of the coupled model is the numerical simulation of laser-tissue interaction.[1]

In dynamic optics, as decribed in [1], a Monte Carlo method is used to simulate laser light scattering and absorption. Heat transport is modeled with a diffusion equation

$$\frac{\partial T}{\partial t} = \frac{1}{c_v}(\nabla \cdot \kappa \nabla T + S) \tag{1}$$

using finite elements.The Monte Carlo simulation of the laser calculates the energy deposition in the spatial zones by the laser, this becomes the source term S in the diffusion equation.

The optical properties of the zones in the computational mesh are altered as the problem progresses by calculating a damage integral based on the zone temperature history.

$$\Omega = \int_0^t k d\tau \tag{2}$$

where

$$k(\tau) = \frac{k_b T(\tau)}{h} \exp\left[\frac{\Delta S}{R} - \frac{\Delta H}{RT(\tau)}\right] \tag{3}$$

k_b is the Boltzmann constant, h is Planck's constant, and ΔS and ΔH are the entropy and enthalpy of reaction, respectively.

Each zone starts with a specified undamaged attenuation coefficient μ_u and varies towards a fully damaged attenuation coefficient μ_d. The quantity Ω is used to generate

[*]Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

[†]Department of Applied Science, University of California@Davis, Livermore Calif.,94550

[‡]X-Division, Lawrence Livermore National Laboratory, Livermore , Calif. , 94550

a new value for the attenuation coefficient μ in each zone. During each time step, after new damage integrals are calculated, the value of μ in each zone is updated:

$$\mu = \mu_u e^{-\Omega} + \mu_d(1 - e^{-\Omega}) \tag{4}$$

2 Domain Decomposition of the Coupled Model

The goal is to decompose the computational domain for a coupled model to realize the best perfomance increase from a parallel computer over a serial computer. The two models being coupled — a Monte Carlo Method and a Finite Element method — will best take advantage of a parallel computer using different domain decompositions.

2.1 Finite Element/Difference methods

The decomposition of the computational grid across processors for a finite element calculation is reasonably well understood.

Methods such as the Recursive Spectral Bisection (RSB) [3] give reasonable decompostions that lead to good load balancing and scaling on a parallel computer.

In the decompostion of the computational grid for a finite element calculation, the amount of computation a processor will do to advance a single zone is fairly constant, hence a decompostion that gives each processor roughly the same number of zones will be approximately load balanced.

Once a finite element calculation has been decomposed across processors, the frequency of message passing to update remote data as well as the size of the messages neeeded is known *a priori.* Methods such as RSB attempt to minimize the number of grid lines cut by the decompostion and so they may claim to minimize the ratio of computation to communication time on a parallel computer.

2.2 Monte Carlo Methods

The approach to domain decomposition for Monte Carlo methods is somewhat different than that for the decomposition in a finite element calculation.

Unlike the finite element calculations discussed above, the amount of work performed in each zone is not necessarily constant. Monte Carlo involves tracking particles in their random walks through the computational grid. Depending on factors such as the problem geometry and the particle sources, the bulk of the work involved with tracking particles can be concentrated in a fraction of the zones in the computational grid. A decompostion that assigns roughly the same number of zones to each processor cannot guarantee a good load balance.

The conventional wisdom on parallelizing Monte Carlo calculations used to model laser light scattering and absorption is decomposition of the particles across the processors. This involves copying the entire computational mesh across each processor, and then dividing the particles to be tracked evenly across the processors. After all processors are finished with all particle tracks, global reductions are used to get the desired quantities (e.g. energy deposited/zone, number of scatters/zone).

This method will fail when the computational grid becomes too large to fit on a single processor. It will also fail in a coupled model — while it provides near-perfect load balance for the Monte Carlo, it will provide no speedup for the finite element calculation.

3 Domain Decomposition of Monte Carlo Calculations

A decomposition method that hopes to give some approximation of load balance must be able to estimate the distribution of the computational work on the computational grid involved in performing the Monte Carlo calculation. The computational effort required will be roughly proportional to the number of particle scatter events that occur in a zone. First, an estimate of the compuational effort required per zone must be made. Then, the domain must be decomposed so that the estimated computational effort is distributed as evenly as possible over the processors.

3.1 The Test Code

We have developed a code that runs on a serial computer that examines different domain decomposition strategies for Monte Carlo calculations.

The grid data consists of the attenuation coefficient, μ, for each zone of a problem geometry.

The code uses a user-specified algorithm to predict the work involved with each zone and decompose the computational grid. The decomposition consists of setting a variable for each zone indicating the processors assigned to that zone.

3.1.1 Running the Problem Once the compuational grid is set up and decomposed, the Monte Carlo computation is executed. The number of particles to be tracked through the problem geometry is a pre-specified parameter. Each particle starts at the particle source. For the problems run in this paper, the source starts each particle at the origin with an initial direction along the x axis.

The distance to the next collision d_{coll} — an exponentially-distributed number with mean $1/\mu_i$ — is calculated, and the particle is advanced a distance d_{coll} in its current direction. A new direction is then randomly generated for the particle. This process continues until the particle travels outside of the problem geometry, or until a specified maximum number of collisions has occured.

Although this is a simplistic treatment of the Monte Carlo method, it allows examination of different domain decomposition strategies.

3.1.2 Timing Simulation The particles in the simulation code are abstractions; they each represent a group of particles. It is assumed that each processor assigned to a zone i will equally share the work to advance a group of particles in zone i from one collision to another. A parameter t_{path} represents the time needed to advance a particle. Timing data are generated by adding t_{path}/P_i to the timer of each processor assigned to the zone i, where the particle last resided. P_i is the number of processors assigned to zone i.

A module was also implemented that attempted to account for the time needed for processors to communicate when needed. Communication would occur when a particle track leaves the domain of a processor and travels to the domain of another processor. Another parameter $t_{message}$ represents the time needed to pass a simulated particle from one processor to another. Every time a particle is advanced, the simulation code compares the processor ownership of the zone entered against the ownership of the zone exited. If the ownerships are the same, no message passing time is added, if they are different, processors losing the particle and processors gaining the particle are charged message passing time.

The test cases discussed in Section 4 were run with $t_{message} = 0$ (ignoring any message passing penalty) and with $t_{message} = 10t_{path}$. The results of each set of runs are summarized in Table 1 and Table 2.

3.2 The Decomposition

The decomposition algorithm used in this paper is fairly elementary. It seeks to divide the computational grid into two roughly comparable halves and to assign processors to each subdomain so that the work per processor is roughly balanced.

If w_i is the estimated computational effort required per zone — proportional to the estimated scatter events per zone — normalized so that the zone with the most estimated computation has $w_i = 1$, the decomposition algorithm will attempt to find w_c so that the sets:

$$\mathcal{H} = \{i | w_i > w_c\} \tag{5}$$

$$\mathcal{L} = \{i | w_i \leq w_c\} \tag{6}$$

have roughly the same number of elements. This was accomplished by starting w_c at $w_{min} = \min w_i$ and comparing the cardinalities, $|\mathcal{H}|$ and $|\mathcal{L}|$. While $|\mathcal{H}| > |\mathcal{L}|$, w_c is increased by adding a fraction of the difference between w_{min} and 1. The size of the step to be taken is is a pre-specified parameter. For the test cases in this paper, each step added 0.01% of the difference between w_{min} and 1.

The value used for w_c is the value that makes the number of zones in $\mathcal{L}$ just greater than the number of zones in $\mathcal{H}$. This will partition the computational mesh into two subdomains. For the problem geometries and estimator functions $w(i)$ considered in this paper, this simple algorithm genrates contiguous domains. More complicated geometries will require an extra step to ensure that the partitioned domains are contiguous.

To assign processors to the subdomains, two more quantites are calculated: the total amount of estimated effort in each subdomain

$$\mathcal{W}_H = \sum_{i \in \mathcal{H}} w_i \tag{7}$$

$$\mathcal{W}_L = \sum_{i \in \mathcal{L}} w_i \tag{8}$$

Since each processor assigned to a subdomain will have a complete copy of the zones in the subdomain, the processors assigned to each subdomain will be able to approximately equally share the computational effort required in the subdomain.

If we let N_P be the number of processors available, N_H be the number of processors assigned to $\mathcal{H}$ and N_L be the number of processors assigned to $\mathcal{L}$, the processors are assigned by initially setting $N_H = N_L = 1$ and comparing the quantites $L_H = \mathcal{W}_H / N_H$ and $L_L = \mathcal{W}_L / N_L$. For each of the remaining processors, if $L_H > L_L$, N_H is incremented, otherwise N_L is incremented. This will balance the estimated effort per processor to within the granularity allowed by N_P. Once N_H and N_L are determined, the test code assigns processors 0 to $N_H - 1$ to subdomain $\mathcal{H}$ and processors N_H to $N_P - 1$ to subdomain $\mathcal{L}$.

4 The Test Problems

Six test cases were considered for each decomposition strategy. The problem geometry is the region given by $(0, 20) \times (-10, 10) \times (-10, 10)$. All test cases use a 3 dimensional, 20x20x20 zone uniform orthogonal grid. In each case, the source of particles was at the origin with all particles initial direction along the x axis.

The quantity varied in the different test cases was the total attenuation coefficient μ_i. When the Monte Carlo method randomly generates the distance to the next collision for a particle currently in zone i, it generates an exponentially distributed random distance with a mean of $1/\mu_i$, i.e., the mean free path $\lambda_i = 1/\mu_i$ [2]. Strictly speaking, the next event a particle undergoes could be a scatter or an absorption. In the problems of interest, however, the scatter coefficient, μ_s, is much larger than μ_a, the absorption coeffcient, so all events are assumed to be scatters.

The first test case , **simple**, had the attenuation coefficient μ constant throughout the mesh so that $\lambda_i = 0.5$ in each zone.

The second test case, **slope1**, varies μ linearly in x from a value of 0.1 for to a value of 10.

The **slope2** test case is analagous to **slope1**, except that μ varies from 10 down to 0.1.

The **path1** case $\mu = 50$ in small region given by $(0,4) \times (-2,2) \times (-2,2)$ around the source with $\mu = 1$ elsewhere. In this case almost no particles will penetrate the small area around the source into the outlying region.

The last two test cases, **plate1** and **plate2**, $\mu = 10$ in the slabs $(0,4) \times (-10,10) \times (-10,10)$ and $(4,8) \times (-10,10) \times (-10,10)$, respectively. $\mu = 1$ outside the slab.

4.1 Predictor Functions Used

Each test case was run in the simulator using three different domain decomposition strategies.

4.1.1 Geometric Decomposition The first strategy, *Geometric Decomposition*, subdivids the grid into as many roughly equally-sized domains as there are processors and assigns a single processor to each domain. This is analogous to the common methods that would be used to decompose a finite element or finite difference problem. The implicit assumption is that the amount of work per zone is constant. This strategy does not use the decomposition discussed in Section 3.2

The next two strategies generate an estimate of the computational effort in each zone and use the decomposition discussed in Section 3.2.

4.1.2 Mean Free Path Decomposition The first of these estimators, *Mean Free Path Decomposition*, uses the attenuation coefficient μ. This strategy assumes that, since the mean free path λ is equal to $1/\mu$, a particle — once in a zone with a large μ — will tend to remain in that zone. This means that the processor handling that zone will have to advance the particle many times before it leaves the zone.

To attempt to account for the fact that zones far from the particle source will tend to have fewer particle scatter events, the mean free path estimate will also be inversly proportional to the square of the distance from the source to the zone. So if μ_i is the attenuation coefficient for zone i, and r_i is the distance from the source to zone i, Mean Free Path Decomposition will have an unnormalized work estimate of

$$W_i = \frac{\mu_i}{r_i^2} \tag{9}$$

which can then be normalized to w_i and passed on to the decompostion method of Section 3.2.

4.1.3 Coarse Grid Decomposition *Coarse Grid Decomposition* creates a coarse version of the problem grid and tracks a small number of particles through the coarse grid. It then extrapolates the computational effort per coarse zone onto the finer zones. The coarse copy of the problem geometry has one zone on each axis for each C_f zones on the actual problem geometry. C_f is a factor that is pre-specified. The attenuation factor μ for each coarse zone is calculated by averaging the μ values for each corresponding zone in the fine mesh.

If n_j is the number of particle scatter events that occured in zone j of the coarse grid after tracking a small number of particles, and $\mathcal{I}_j$ is the set of zones on the fine mesh that correspond to zone j in the coarse mesh, then the unnormalized work estimate W_i will be defined as

$$i \in \mathcal{I}_j \Rightarrow W_i = n_j \tag{10}$$

Again W_i can be normalized to w_i and passed to the decomposition method of Section 3.2.

Each test case was run using coarse grid decomposition with coarseness factors $C_f = 2$ and $C_f = 4$.

5 Results

The results of the load balancing studies for the decomposition strategies are summarized in Table 1 and Table 2. The computation times for each processor calculated by the simulator code were normalized so that the processor with the longest computation time has a time of 1. The column labeled *mean* has the mean of the simulated computation times over processors, the *variance* column contains the variance of the normalized times over the processors. Mean times close to one indicate test cases where the decomposition yielded roughly load balanced caclulations.

TABLE 1
Load Balancing Simulation Results Without Messaging

	Geometric		Mean Free Path		2:1 Coarse Grid		4:1 Coarse Grid	
Problem	*mean*	*variance*	*mean*	*variance*	*mean*	*variance*	*mean*	*variance*
simple	0.76	0.11	0.31	0.10	0.63	0.01	0.97	4.3e-3
slope1	0.77	0.08	0.20	0.06	0.34	0.03	0.96	8.8e-3
slope2	0.85	0.08	0.23	0.07	0.99	1.3e-4	0.99	1.6e-3
path1	0.86	0.11	0.24	0.18	0.93	0.06	0.94	0.06
plate1	0.87	0.10	0.27	0.16	0.95	0.04	0.94	0.05
plate2	0.85	0.09	0.42	0.08	0.93	3.e-4	0.82	2.0e-3

TABLE 2
Load Balancing Simulation Results With Messaging

	Geometric		Mean Free Path		2:1 Coarse Grid		4:1 Coarse Grid	
Problem	*mean*	*variance*	*mean*	*variance*	*mean*	*variance*	*mean*	*variance*
simple	0.25	0.074	0.97	7.98e-4	0.12	0.050	0.37	0.056
slope1	0.25	0.074	0.97	7.98e-4	0.12	0.050	0.37	0.056
slope2	0.26	0.087	0.91	1.84e-3	0.99	5.82e-5	0.77	0.023
path1	0.24	0.16	0.75	0.19	0.96	0.018	0.94	0.044
plate1	0.26	0.15	0.84	0.11	0.50	0.050	0.99	1.17e-5
plate2	0.36	0.089	0.94	8.20e-4	0.38	0.086	0.52	0.032

6 Conclusions and Future Work

The Geometric decomposition (Section 4.1.1) was generally the worst performer, yielding highly unbalanced work loads. This was expected, as it considered a zone far from the source as equally important to a zone adjacent to the source.

Of the strategies that attempted to make some prediction of the distribution of the distribution of the work, the Coarse Grid decomposition (Section 4.1.3) provides better results for most of the test cases considered when message passing time is ignored. If it is assumed that $t_{message} = 10t_{path}$, the Mean Free Path Decomposition generates the best results.

The simulation code has shown some promise as a testbed that allows quick prototyping and testing of domain decomposition algorithms for Monte Carlo transport calculations. The results suggest that both domain decomposition strategies considered have promise.

The next task planned is the implementation of a code to allow the same elementary Monte Carlo calculations discussed in this paper to run on a parallel computer such as the T3D at Lawrence Livermore National Laboratory. The decomposition strategies examined in this paper will be implemented and tested in parallel calculations.

References

[1] Richard A. London, Micheal E. Glinsky, George B. Zimmerman, David C. Eder, Steven L. Jaques, *Coupled Light Transport-Heat Diffusion model for Laser Dosimetry with Dynamic Optical Properties*, SPIE Proceedings, Laser-Tissue Interaction VI, San Jose, CA, Feb 1995, v. 2391, p435

[2] S. A. Prahl, M. Keijzer, S. L. Jaques, A. J. Welch, *A Monte Carlo Method of Light Propagation in Tissue*, SPIE Institute Series v. IS 5 (1989)

[3] Bruce Hendrickson, Robert Leland, *An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations*, Technical Report SAND92-1460, Sandia National Laboratories, 1992.